FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Technical Report

Performance of global communications using BLACS on the SMP cluster ZAMpano

Inge Gutheil

FZJ-ZAM-IB-2003-06

April 2003

(last change: 30.04.2003)

Performance of global communications using BLACS on the SMP cluster ZAMpano

Inge Gutheil

John von Neumann-Institut für Computing
Zentralinstitut für Angewandte Mathematik
Forschungszentrum Jülich GmbH
April 30, 2003

Abstract

BLACS (Basic Linear Algebra Communication Subprograms) are the communication layer for ScaLAPACK, a Linear Algebra Package for parallel computers using message-passing for inter-processor communication. Version 1.7 of ScaLAPACK uses the ratio of the transfer cost per element for the combine sum to one processor and the broadcast operation for tuning the Level 3 PBLAS (Parallel Basic Linear Algebra Subprograms). We measured performance of broadcast, global combine and point-to-point communications using BLACS on the SMP cluster ZAMpano.
It turned out that the default value CBRATIO for the above ratio given in the public domain version of ScaLAPACK 1.7 is a reasonable choice for ZAMpano.

1 Introduction

The Basic Linear Algebra Communication Subprograms BLACS [2] are the communication layer for ScaLAPACK and PBLAS [4]. They are based on MPI and support point-to-point communications as well as broadcast and global combines for matrices as these operations are commonly used in parallel Linear Algebra packages. More information about the BLACS can be found in [3]
ZAMpano [1] is a cluster of eight 4-processor Pentium III nodes with Linux operating system (Suse Linux 7.2) and MPI installed as MPICH-GM 1.2.1..7b on GM 1.5.1 using Myrinet between nodes and Myrinet or shared-memory for communication on one node. The transfer rate achieved for message passing via GM is up to 120 MB/sec between nodes and up to 200 MB/sec for shared-memory intra-node communications at a latency of approximately 15 μsec.
We measured times for broadcasts and global sum to one processor with different numbers of processors and message lengths from zero-Byte messages ($N = 0$) to $N = 20.000.000$ Double Precision words which is about 160 MB. For intra-node communication we compared both, myrinet and shared-memory. From the times measured we computed the transfer rates in MB/sec.
Broadcast and global combine measurements were started with the number of nodes and processors per node (usually 4) we had reserved (=NPROCS processors) and then times were measured for USED_PROCS=NPROCS, NPROCS/2, ... until at least 2 processors took part in the communication. Thus we got measurements for USED_PROCS=32, 28, 24, 20, 16, 14, 12, 10, 8, 7, 6, 5, 4, 3, and 2 processors. Broadcast means that one processor is sending to USED_PROCS-1 receiving processors and global combine to one processor means USED_PROCS-1 processors are sending to one processor combining the results.
For completeness we also measured time and transfer rate for point-to-point communication with a ping-pong test between two processors on one node with shared-memory communication and myrinet communication as well as between two processors on different nodes.
The programs used for performance measurements are based on those described in [6].

2 Performance of Broadcast

2.1 How to measure broadcast performance

We choose two ways to measure performance of broadcasts - broadcasts in a cycle and measuring a single broadcast - but they both do not deliver exact results.
Figures 1 to 6 show times spent in msec. Many measurements were done for each number of processors and each message length, and the deviations from the shortest and the longest times were very large when measuring single broadcasts. We found out, however, that there were only a few very short and a few very long times, most

were close to the median. Thus for each message length N (i.e. vectors of length N Double Precision or Integer values are sent) we show at least two values in figures 1 to 4, the shortest and the longest time used, for measurements of single broadcasts the median is shown, too.

1. Broadcasts in a cycle:
   First all processors are syncronized by a barrier call and timing is started. Then for K=0 to USED_PROCS-1 processor K is sender and all other processors are receivers. For small messages this is repeated LOOPS=100/USED_PROCS times, for messages of lengths $N > 100\ 000$ (i.e. more than 100 000 Double or Integer values are sent) LOOPS=LOOPS/2, for $N > 1\ 000\ 000$ LOOPS is again divided by two, but it is made sure that LOOPS is at least 1. After all those broadcasts the processors again are synchronized by a barrier and timing is finished. The elapsed time for one broadcast is then taken to be total time divided by (LOOPS*USED_PROCS). The problem with this measurement is that for a large number of processors this could underestimate the time for broadcast. Processor 1 could already start the second broadcast just after having sent the first message for the last time without waiting for the last processor to have received it. To avoid this underestimation we did the measurements again this time starting the cycle with processor USED_PROCS-1 down to processor 0, and indeed for processor numbers $NP >= 4$ and message lengths smaller than $N = 3000$ (12KB) the times measured were higher with shared-memeory as well as with myrinet communication (see figures 1 and 2 and figure 8).

   When messages become larger than $N = 50000$ the differences in times between the different measurements are no longer visible (see figures 5 and 6). For some processor numbers (expecially $NP = 20$), the transfer rates measuring a single broadcast still differ from those measured with broadcast cycles, but the differences are small.
   For two processors the broadcast in a cycle is the same as the ping-pong timing for point-to-point communication, in that case this method surely is the best one (see figures 3 and 4).

   Figures 1 to 4 show that the difference between the lowest and the highest execution time for a broadcast of the given length measured in a cycle is very small.

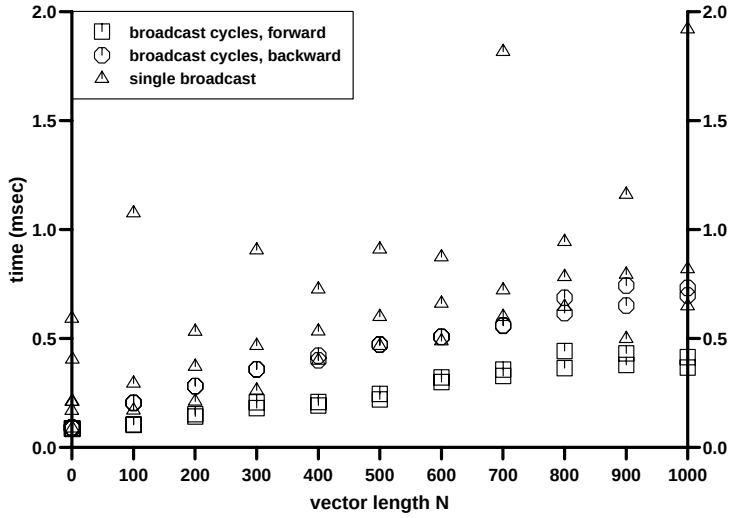

Figure 1: Execution times for broadcasts in a cycle, sender goes from processor 0 to processor USED_PROCS-1 (broadcast cycles, forward) and from processor USED_PROCS-1 down to processor 0 (broadcast cycles, backward) compared to measurement of a single broadcast, USED_PROCS=32, vector of length $N = 0, \ldots, 1000$ Double Precision values, communication within each node using shared memory, between nodes using myrinet.

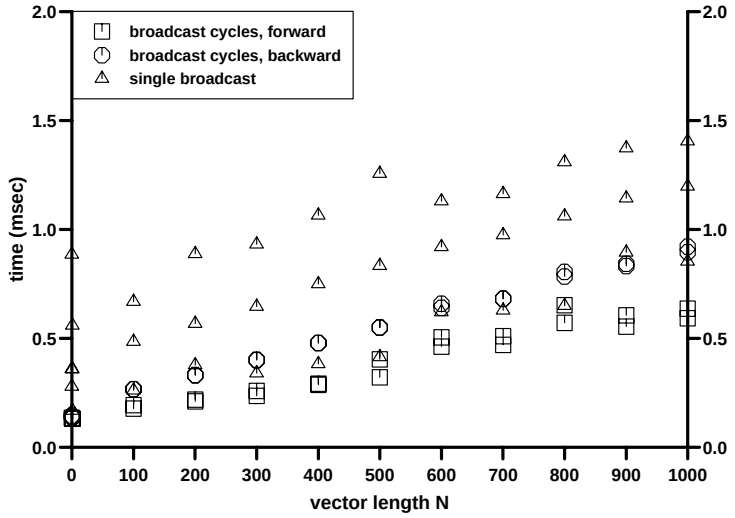

Figure 2: Execution times for broadcasts in a cycle, sender goes from processor 0 to processor USED_PROCS-1 (broadcast cycles, forward) and from processor USED_PROCS-1 down to processor 0 (broadcast cycles, backward) compared to measurement of a single broadcast, USED_PROCS=32, vector of length $N = 0, \ldots, 1000$ Double Precision values, communication always using myrinet.

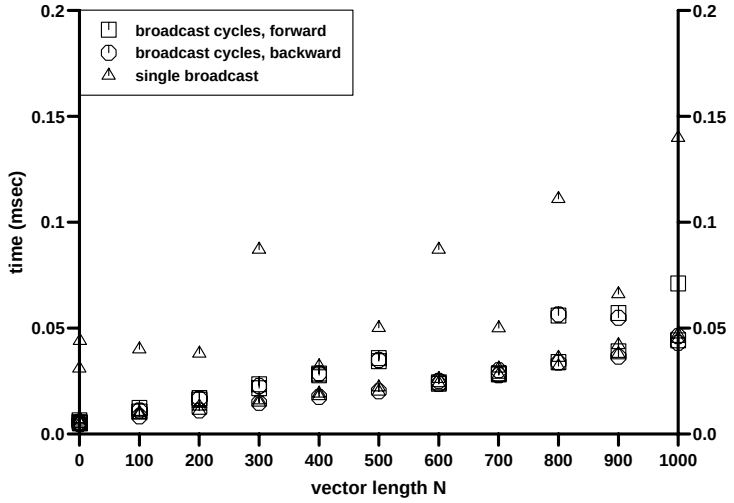

Figure 3: Same as figure 1 but with USED_PROCS=2.

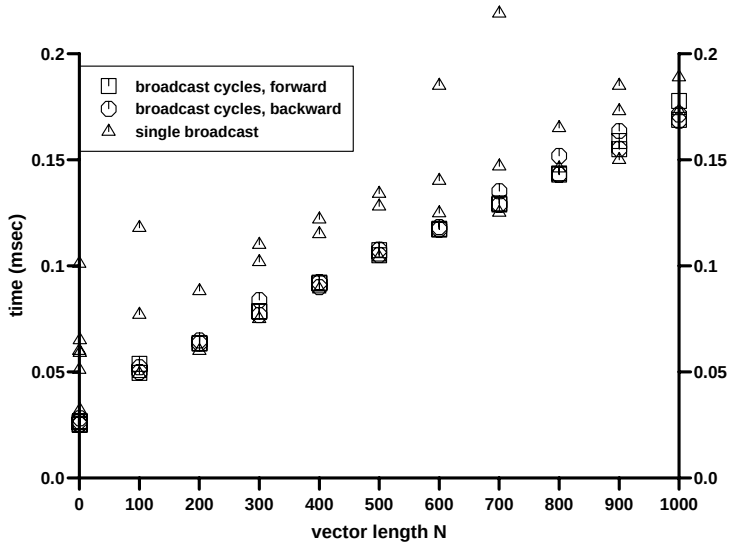

Figure 4: Same as figure 2 but with USED_PROCS=2.

2. Timing a single broadcast:
   This only delivers trustable results for many processors and long messages, because the execution times for a single broadcast are in the range of some milliseconds even for 10000 Double Precision values (80 KB) and 8 processors. The only way to get significant results is to repeat the broadcast, measure the time for many broadcasts and divide it by the number of measurements, but this cannot be done without synchronization between the broadcasts, and this means that synchronization times will be measured, too.
   We decided to do the following:
   Synchronization, start timing, broadcast from processor 0 to all other processors, stop timing on all processors and find out the times each processor needed from the barrier until it had received the broadcast message (for the sender the time until it had finished sending the message). The relative difference between the time on the processor with the longest and one with the shortest execution time is very large for times less than some milliseconds, for larger times the relative difference becomes small. The largest time is taken as the time for the broadcast as this would be the time for the broadcast if all processes had left the barrier at the same moment. Again there is a problem of underestimating the time for one broadcast if the last processor leaving the barrier is the last processor getting the message.

   Figures 1 to 4 show that the median of the times measured with single broadcasts is higher than the times measured with cycles for small messages and the shortest execution times measured with single broadcasts are about the same as the times measured with cycles.

From figures 5, 6, and 7 it can be seen that for messages longer than about $N = 10000$ Double precision words (80 KB) all methods for measuring broadcast times with 32 processors deliver almost the same results.

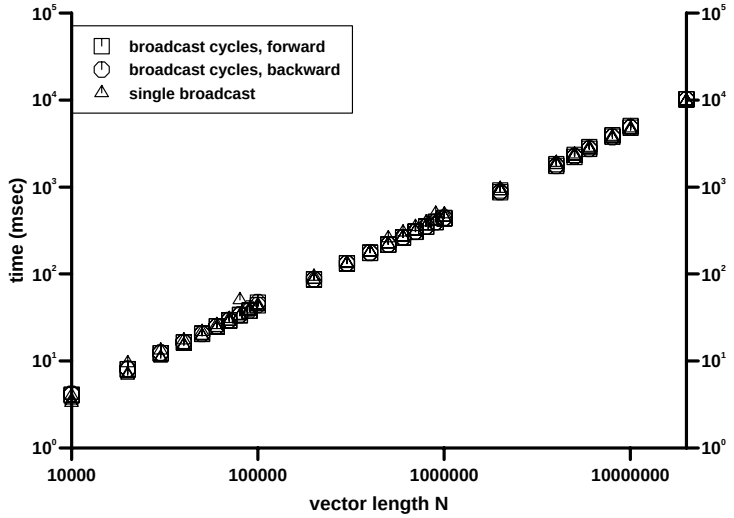

Figure 5: Execution times for broadcast, all methods, USED_PROCS=32, Double Precision vectors, communication using shared-memory within each node and myrinet between nodes.

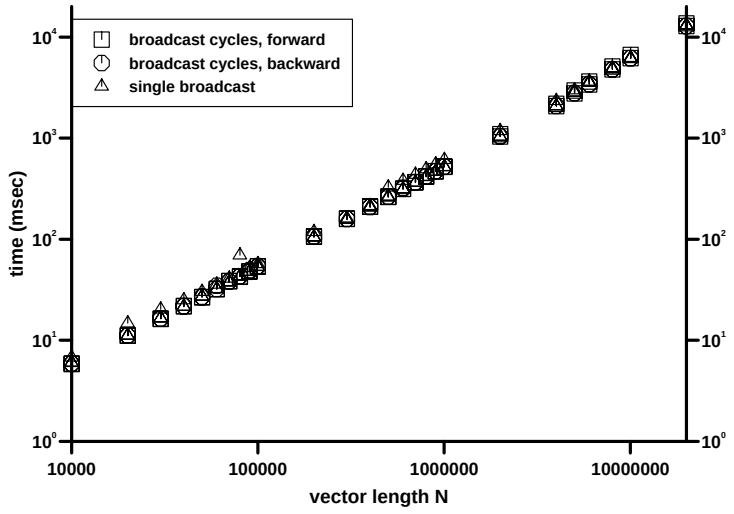

Figure 6: Execution times for broadcast, all methods, USED_PROCS=32, Double Precision vectors, communication always using myrinet.

2.2 Results of broadcast measurements

First of all we can state that transfer times for N Double Precision values are about twice as high as transfer times for N Integer values for all message lengths and processor numbers, just as we expected it. The transfer rates achieved are the same for Double Precision as for Integer, and the transfer rates are best suited to show the results of performance measurement for communications. Figures 7, 8, 11, and 12 thus will show transfer rates for double precision values. Only the medians of the measurements are shown.

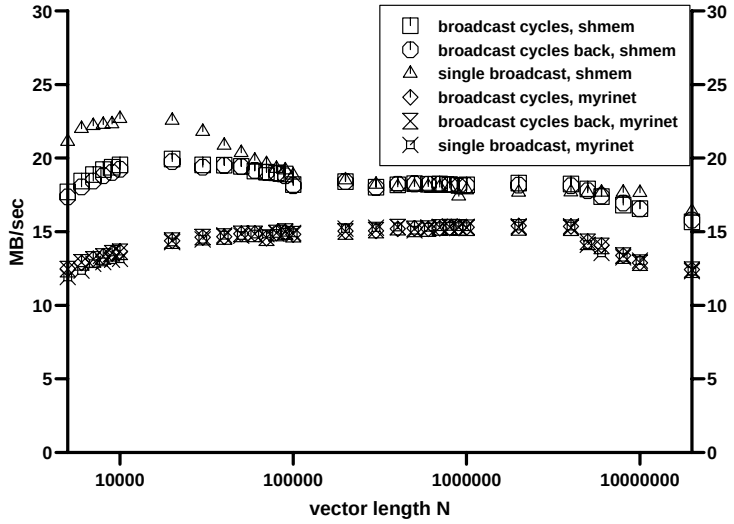

Figure 7: Transfer rates for broadcast, all methods, USED_PROCS=32, large Double Precision vectors, myrinet means that myrinet is used for all communications, shared-memeory means that within each node shared memory is used.

Figure 7 shows basically two different "curves", the one for communication using shared-memory within one node, and the one for communication using always myrinet. The latter shows an about 5 MB/sec smaller transfer rate but overall almost the same trend. With smaller message lengths other factors, especially the way measurement is done, dominate and the trends are no longer clear (see figure 8).

Similar behaviour as shown in figure 7 with $NP = 32$ processors (=8 nodes) can be seen for all processor numbers $NP \geq 5$ (at least 2 nodes). The more nodes are used the smaller become the differences between using shared-memeory within each node and using myrinet even within a node.

The highest transfer rates are achieved for messages of lengths between $N = 10000$ and $N = 50000$ Double Precision words (80 KB to 400 KB) and they reach almost 20 MB/sec for broadcasts with 32 processors when shared-memeory is used within each node. Then the transfer rates decrease to about 18 MB/sec for messages of about $N = 100000$ to $N = 3000000$ (800 KB to 24 MB). When only

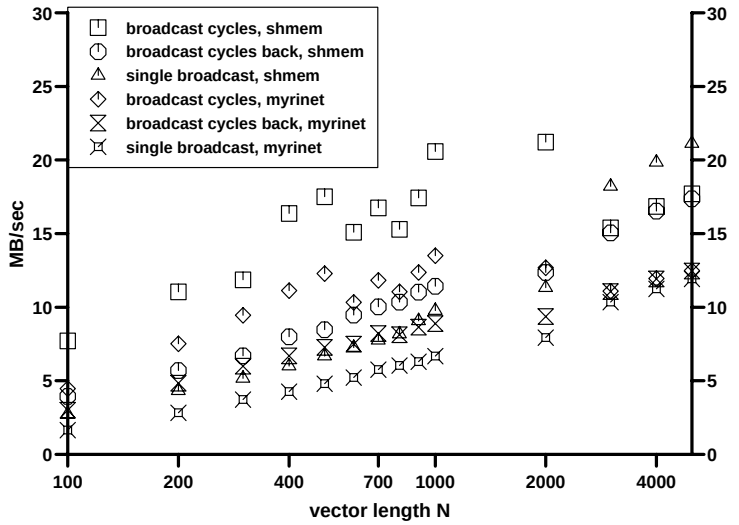

Figure 8: Transfer rates for broadcast, all methods, USED_PROCS=32, short Double Precision vectors, myrinet means that myrinet is used for all communications, shared-memeory means that within each node shared memory is used.

myrinet communication is used the transfer rates are best for messages between $N = 40000$ and $N = 3000000$ Double Precision words (320 KB to 24 MB) and reach more than 15 MB/sec there.

For messages of length $N = 5000000$ and more (40 MB and more) the transfer rate for broadcast decreases with any number of processors using myrinet communication everywhere. Using shared-memeory communication within each node this decrease is seen for more than 5 processors and it is slighter than with myrinet communication when less than eight processors (no more than two nodes) are taking part in the broadcast.

Communication latency reduces the transfer rates for small messages. The measured time to broadcast a zero-length message to 32 processors was about $0.9 \cdot 10^{-4}$ sec using shared-memeory within each node and about $1.4 \cdot 10^{-4}$ sec using myrinet communication. This means that for a message of length $N = 1000$, which is 8000 Bytes, the measured time of about 0.7 msec using shared-memeory within each node (see figure 1) consists of $0.9 \cdot 10^{-4}$ sec latency and $6 \cdot 10^{-4}$ sec transfer time, which is a transfer rate of about $8 \cdot 10^3 \text{MB}/6 \cdot 10^{-4} \text{sec} \approx 13$ MB/sec instead of 11 MB/sec with the total time. The total time of 0.9 msec measured using myrinet (see figure 2) consists of $1.4 \cdot 10^{-4}$ sec latency and $7.6 \cdot 10^{-4}$ sec transfer time, which is a transfer rate of about $8 \cdot 10^3 \text{MB}/7.6 \cdot 10^{-4} \text{sec} \approx 10.5$ MB/sec instead of 9 MB/sec with the total time.

The transfer rates indicate that the broadcast cannot be done by sending the mes-

sage around from one processor to the next one. With that kind of communication the transfer rate could be at most $\frac{1}{31}$ of the peak transfer rate which would be about $\frac{120}{31} \approx 4$ MB/sec with myrinet communication between all processors. The maximum transfer rate however is about 15 MB/sec with myrinet communication. This means that no more than 8 transfers can be done sequentially, the other messages must be sent in parallel.
This suggests that the broadcast is done in stages (see figure 9) just as we found out for CRAY T3E in [6], although the steps are not as significant as on CRAY T3E. Figure 10 shows that the communication times have little "jumps" when sending to more than 1, more than 3, more than 7, and more than 15 processors, whereas they remain almost constant when sending to 6, 11, 13, and 15 processors which is always done in 4 steps.

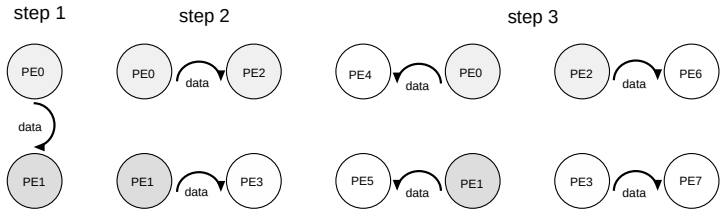

Figure 9: Propagation of data, broadcasts

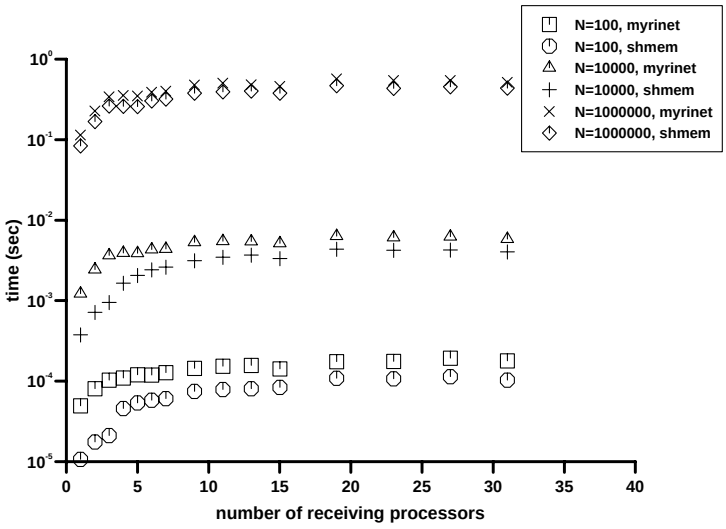

Figure 10: Communication times for broadcasts of messages of three different lengths, $N = 100,\ N = 10000,$ and $N = 1000000$ Double Precision words to different numbers of processors, both kinds of communication, only the shortest execution times measured for broadcast in a cycle forward are shown.

A significant decrease in the transfer rate of messages with lengths up to approximately $N = 70000$ can be seen when 5 instead of 4 processors are taking part in the broadcast using shared memory for communication within one node (see figure 11) as now at least one processor is outside the node and can only be reached via myrinet comunication. When the messages become larger this decrease is much smaller or even vanishes. The transfer rates of shmem broadcast within one node of 4 processors decreases dramatically from 80 MB/sec for messages of length between $N = 5000$ and $N = 10000$ to about 30 MB/sec for messages larger than $N = 500000$ (see figure 11).

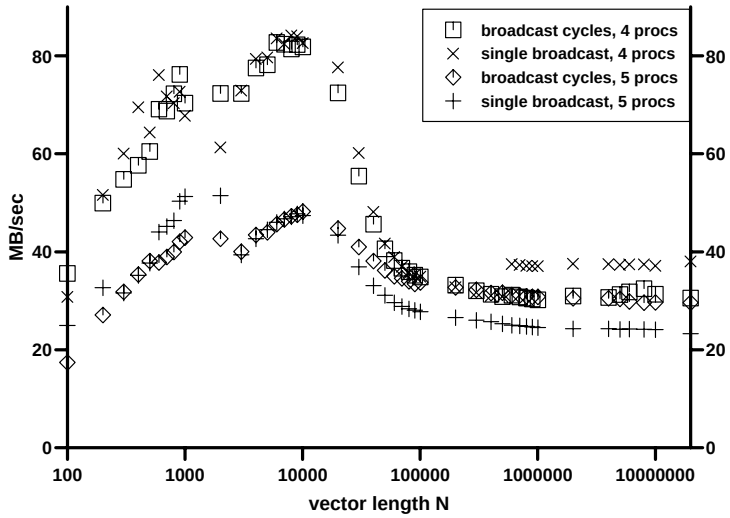

Figure 11: Transfer rates for broadcast, 4 and 5 processors, communication using shared-memeory on a single node. As there is no significant difference between measuring broadcasts in a cycle forward or backward, only forward is shown.

We cannot explain the strange phenomenon that with 4 processors doing broadcast using shared-memeory within one node the measurement of the execution time of a single broadcast for large messages shows much higher transfer rates than measurement of broadcast cycles, whereas with 5 processors taking part the result is the other way round. An explanation could be the fact that when measuring a single broadcast always processor 0 is the sender, so the sending pattern is always the same, whereas in cycles the sending patterns differ as the senders are always different.

Using myrinet between all processors shows a more steady behaviour. Measuring a single broadcast and broadcast cycles gives almost the same result when sending within one node (to 3 processors) and always gives a smaller communication rate when sending to 4 processors. The difference between broadcast to 3 processors and to 4 processors is almost constant for all problem sizes (see figure 12).

The jumps in execution times and thus also in transfer rates when the number of processors exceeds a power of two can also be seen from the transfer rates of the

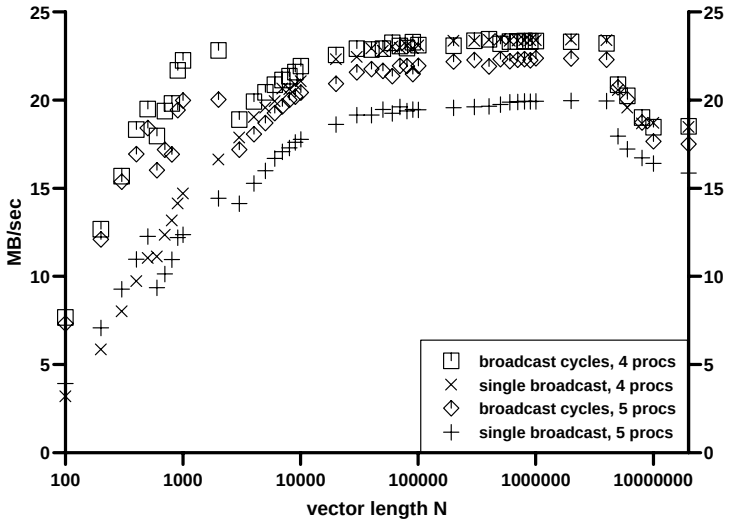

Figure 12: Transfer rates for broadcast, 4 and 5 processors, communication using myrinet even within a node. As there is no significant difference between measuring broadcasts in a cycle forward or backward, only forward is shown.

other processor numbers. For messages of lengths $N = 80000$ to $N = 3000000$ (320 KB to 24 MB) using myrinet communication even within a node the transfer rate is $\approx$ 20 MB/sec for 5 to 8 processors, $\approx$ 17 MB/sec for 10 to 16 processors and $<$ 15 MB/sec for 20, 24, and 28 processors taking part. For 32 processors it is slightly higher than for 20 to 28 processors.

Using shared-memeory communication within each node the effect of the broadcast steps by powers of two is overlapped by the shared-memeory effects which become smaller with an increasing number of processors. Thus there is a steady decrease in the maximum transfer rate for messages longer than $N = 50000$ (400 KB) from 30 MB/sec using 5 processors to 25 MB/sec using 8 processors. For 10 to 16 processors it is always around 20 MB/sec and for 20 to 32 processors it is $\approx$ 18 MB/sec.

3 Performance of Point-to-Point Communication

Point-to-point communication was measured using the so-called ping-pong test: a message is sent from one processor to the other one and back, LOOPS times, the time for these send/receive operations is measured, and the time for one point-to-point send is taken as elapsed time divided by (2*LOOPS).

There are three different cases:

1. The two processors are in one node and communicate using shared-memeory.
2. The two processors are in one node and communicate using myrinet.
3. The two processors are in two different nodes, they can only use myrinet for communication.

All measurements were done several times and the median is shown in the following figures. For some message sizes there were large deviations from the median which we cannot explain. In most cases, however, all values were close together. The broadcast values shown are measured with broadcast cycles forward which is the same as ping-pong.

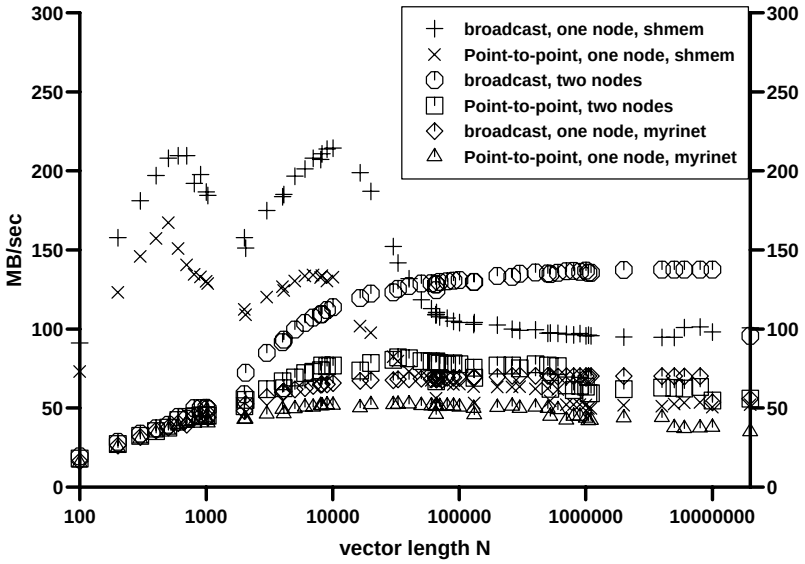

Figure 13: Transfer rates for BLACS point-to-point communication, broadcast and send/receive, only medians are shown.

To our great surprise the transfer rates for point-to-point communication were smaller than the rates for broadcast to one processor in all cases (see figure 13).

We measured point-to-point communication and broadcast to one node with MPI and the performance of broadcast to one node and point-to-point communication with MPI is almost the same in all cases (see figure 14).

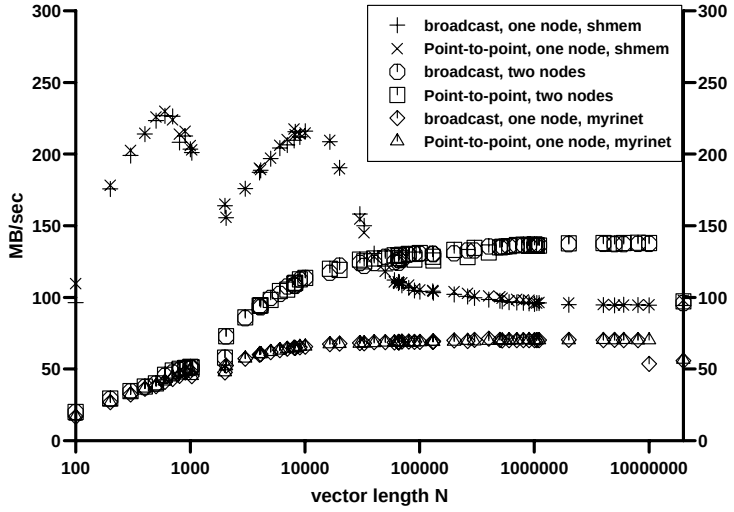

Figure 14: Transfer rates for MPI point-to-point communication, broadcast and send/receive, only medians are shown.

Figure 15 shows that there is almost no difference in transfer rates between BLACS and MPI broadcasts to one node. Thus we think that BLACS broadcast uses MPI broadcast directly whereas BLACS point-to-point communication does something more than just MPI point-to-point communication.
From this we would suggest to use BLACS broadcast wherever possible even if there is only one receiver. But as all processors in a context must take part in a broadcast operation, for each point-to-point communication a new BLACS context is needed, and we did not measure the costs of starting a new BLACS context.

The next thing to observe is that communication using shared-memeory within one node does not always deliver the highest transfer rates. The transfer rates using shared-memeory have two peaks, one with messages of $N = 500$ to $N = 700$ (40 KB to 56 KB) and one for message lengths of $N = 9000$ to $N = 10000$ (72 KB to 80 KB). In both cases the transfer rates are higher than 200 MBytes/sec. This is even more than we could expect. Between these two peaks there is a minimum at $N \approx 2000$ (16 KB) where the transfer rate is only about 155 MBytes/sec.
For messages larger than $N = 10000$ (80 KB) the transfer rate for shared-memeory communication decreases and for $N \geq 70000$ (560 KB) it becomes even smaller than the transfer rate between two different nodes. This rate increases steadily from less than 20 MBytes/sec for $N = 100$ (almost only the latency, no

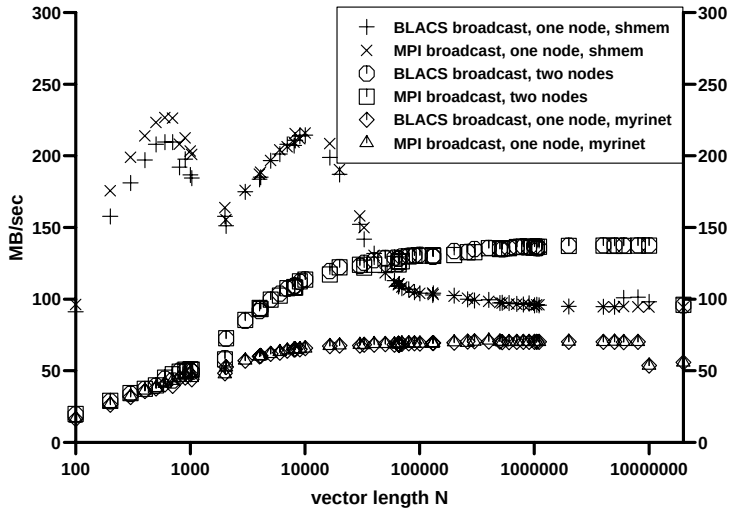

Figure 15: Transfer rates for BLACS and MPI point-to-point communication using broadcast, only medians are shown.

transfer costs) to 139 MBytes/sec for $N = 10000000$ (=80 MB in a little more than 1/2 sec). Again the transfer rate reached is higher than expected.

The two performance peaks with one minimum between them at $N \approx 2000$ can be seen for all broadcasts within one node using shared-memeory and even for broadcasts with one node and one extra processor using shared-memeory within the node (see figure 11). The decrease of the transfer rate using shared-memeory within one node for messages longer than $N = 10000$ (80 KB) can be seen for broadcasts with all numbers of processors.
A possible explanation for the two peaks can be a change in the MPI protocol at the size of $N = 2000$ which is 16 KBytes. This is the same size where shared-memory communication is very slow compared to message sizes close to this size and communication between two different nodes speeds up in a jump.
The fact that communication from one node to another one is faster than shared-memory communication within one node for large messages can be explained by the relatively small memory bandwidth within one node. In the case of shared-memory communication within one node two processors are reading and writing to the same memory at the same time whereas with inter-node comunication only one processor per node is reading from or writing to the node's memory.

4 Performance of Global Combine Sum to one Processor

As global combine to one processor is almost the reverse operation of broadcast - now $NP - 1$ nodes are sending and the result is combined (here summed up) on one node - there are the same problems of measuring execution times of global combine as for broadcast.

We did the same kinds of measurement, combine in a cycle, now the receiver goes from processor 0 to processor $NP - 1$ or vice versa and we tried to measure a single global combine.
Like figures 1 and 2 figures 16 and 17 show the minimum and maximum times measured and for the single global combine also the median of the measurements.

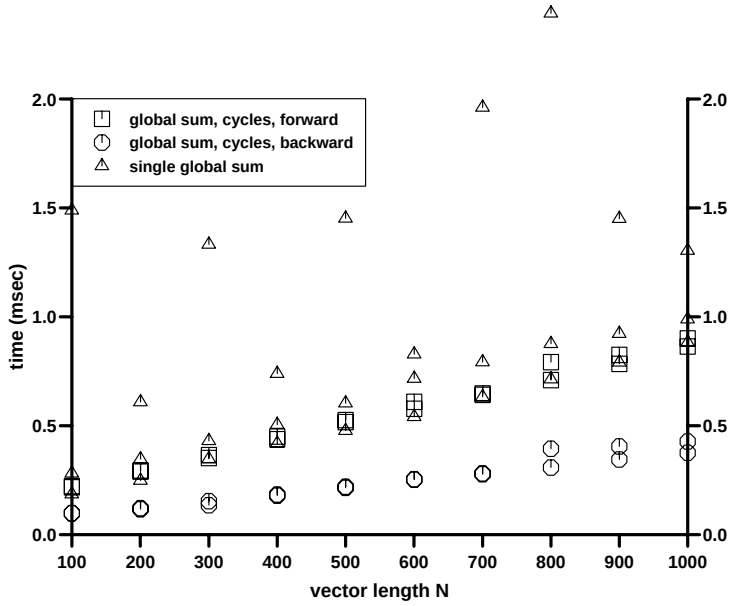

Figure 16: Execution times for global sum in a cycle, receiver goes from processor 0 to processor USED_PROCS-1 (global sum, cycles, forward) and from processor USED_PROCS-1 down to processor 0 (global sum, cycles, backward) compared to measurement of a single global sum, USED_PROCS=32, vector of length $N = 100, \ldots, 1000$ Double Precision values, communication within each node using shared-memory, between nodes using myrinet.

Figures 16 and 17 look rather similar to figures 1 and 2 with the exception that now the backward cycles are faster than the cycles forward for $N \leq 2000$. An explanation can be the steps of the global sum which are reverse to the steps of broadcast as can be seen in figure 18. The first processors to finish global sum to processor 0 are processor $\frac{NP}{2}$ to $NP - 1$, they only have to send their elements to processors 0 to $\frac{NP}{2} - 1$. The last one to finish is processor 0, it has to receive

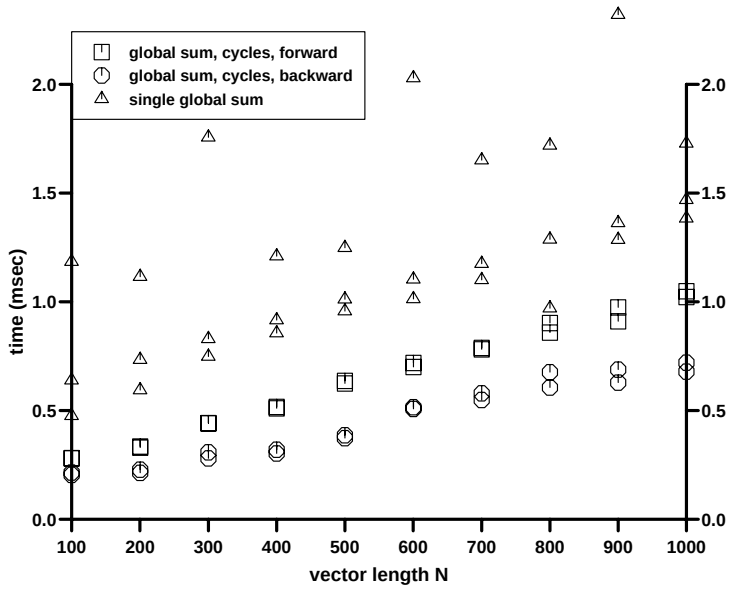

Figure 17: Execution times for global sum in a cycle, receiver goes from processor 0 to processor USED_PROCS-1 (global sum, cycles, forward) and from processor USED_PROCS-1 down to processor 0 (global sum, cycles, backward) compared to measurement of a single global sum, USED_PROCS=32, vector of length $N = 100, \ldots, 1000$ Double Precision values, communication using myrinet even within each node.

and add results several times, the last time it adds the result from processor 1. If the next global sum is done to processor 1, the first processors to send now are processors $\frac{NP}{2} + 1$ to processor 0, and as processor 0 is the last to finish the first sum, the next global sum can only start when the first one is finished. If on the other hand the next global sum is accumulated to processor $NP - 1$ (cycles backward) the first processors to send data for the second sum are processors $\frac{NP}{2} - 1$ to processor $NP - 2$. These processors have finished the first global sum long before the first processor has got the result. This fact seems to lead to the fast times when measuring backward cycles for global sum with short messages.
We cannot explain why for $N \geq 3000$ the times measured with cycles forward are sometimes shorter than the times measured with cycles backward.
Due to the fact that we still measured communication of Double Precision and Integer we had allocated too much memory to measure the combine times for 20000000 Double Precision values in cycles. Only measurements for single combines were done for such large vectors. The value for this measurement is not reliable because paging effects could be included.
The deviations from the median are larger than the deviations of broadcast measurements when single global sums are measured. There is the strange phenomenon that the last time measured in a series of single global sums of the same

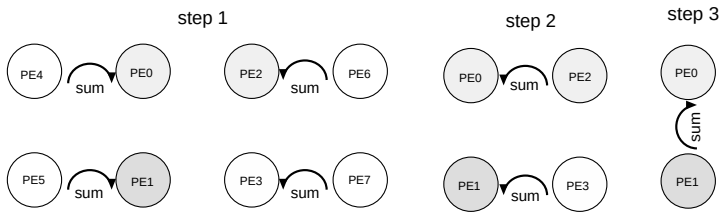

Figure 18: Propagation of data, global sum

size is up to 10 % longer than the other times in that series if a global sum of a larger size is measured thereafter. If we just ignored the first and last measured value of a series of measurements of a single global sum of the same size the deviations from the median would be much smaller.
For large messages the deviations from median are still larger for global sum than for broadcast, but the overall behaviour is the same (see figures 19 and 20).

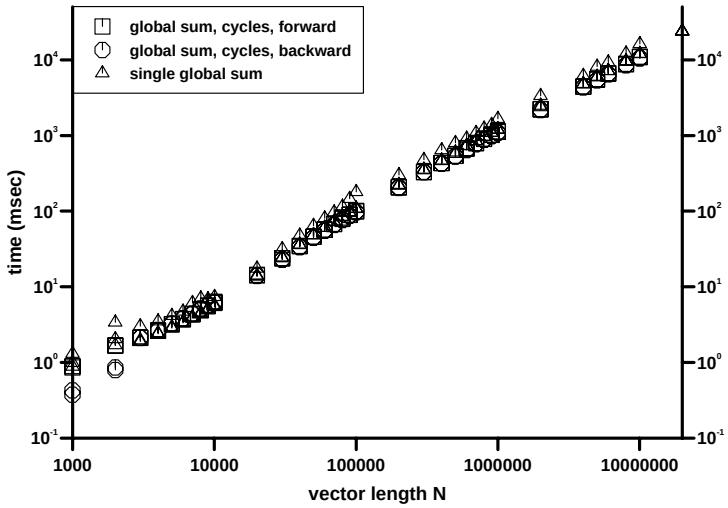

Figure 19: Execution times for global sum in a cycle, receiver goes from processor 0 to processor USED_PROCS-1 (global sum, cycles, forward) and from processor USED_PROCS-1 down to processor 0 (global sum, cycles, backward) compared to measurement of a single global sum, USED_PROCS=32, vector of length $N = 100, \ldots, 20000000$ Double Precision values, communication within each node using shared memory, between nodes using myrinet.

To show and compare a large range of vector lengths we introduce a ”combine rate” measured in MB per second (MBps). A combine rate of 8 MBps means that 1 million Double Precision numbers can be summed up from NP processors to one processor in one second. This includes transfer and addition. With these combine rates the difference in performance between shmen and myrinet communication can be seen much better than with times shown on a logarithmic

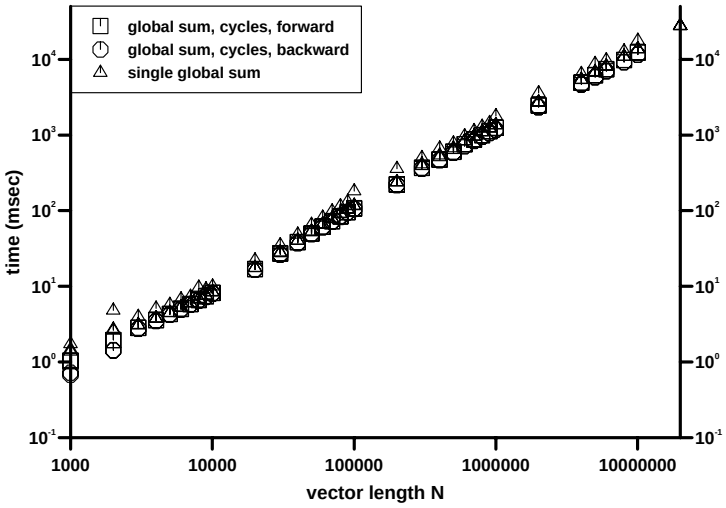

Figure 20: Execution times for global sum in a cycle, receiver goes from processor 0 to processor USED_PROCS-1 (global sum, cycles, forward) and from processor USED_PROCS-1 down to processor 0 (global sum, cycles, backward) compared to measurement of a single global sum, USED_PROCS=32, vector of length $N = 100, \ldots, 20000000$ Double Precision values, communication always using myrinet.

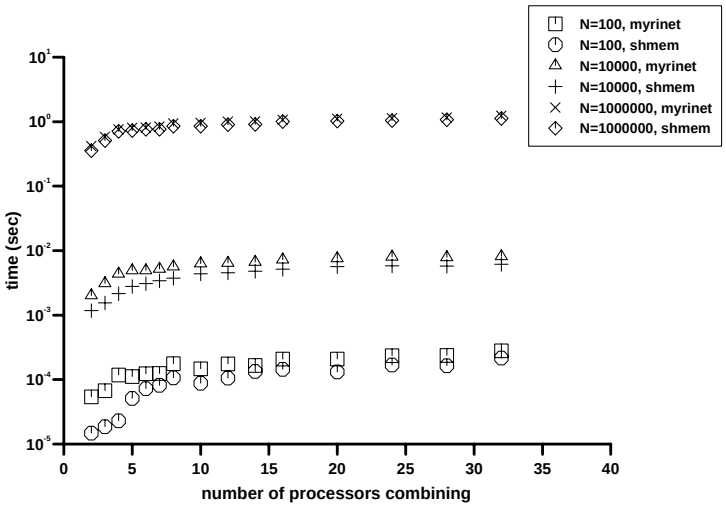

Figure 21: Execution times for global sum of vectors of three different lengths, $N = 100$, $N = 10000$, and $N = 1000000$ Double Precision words with different numbers of processors, both kinds of communication, only the shortest execution times measured for combine in a cycle forward are shown.

timescale. Like the transfer rates for broadcasts the combine rates for global sum vary only in a very limited range.

The ”curves” of the combine rates are very similar to those of transfer rates for broadcasts, but the differences between myrinet communication overall and shared-memeory communication within one node are smaller for global combine whereas the differences between different kinds of measurement are much higher (see figure 22). Especially for small messages ($N \leq 2000$, i.e. up to 16 KB) the combine rate measured with backward cycles is much too high. For $N \geq 3000$, i.e. for 24 KB and more) the combine rates of both cycles are close together and only the rates when measuring a single combine are different and for $N \geq 30000$ (240 KB) smaller than those measured with cycles.
The two performance peaks for small N with a minimum at $N = 2000$ can be seen again on a single node if shared-memeory is used (figure 23). The performance decrease for $N > 10000$ (80 KB) using shared-memeory on each node is not as strong as for broadcast. In contrast to broadcast the combine rate using myrinet even within a node also decreases for $N > 10000$ (figure 22). We thus expect the ratio of time for global sum to time for broadcast to increase for more than 10000 Double precision values.

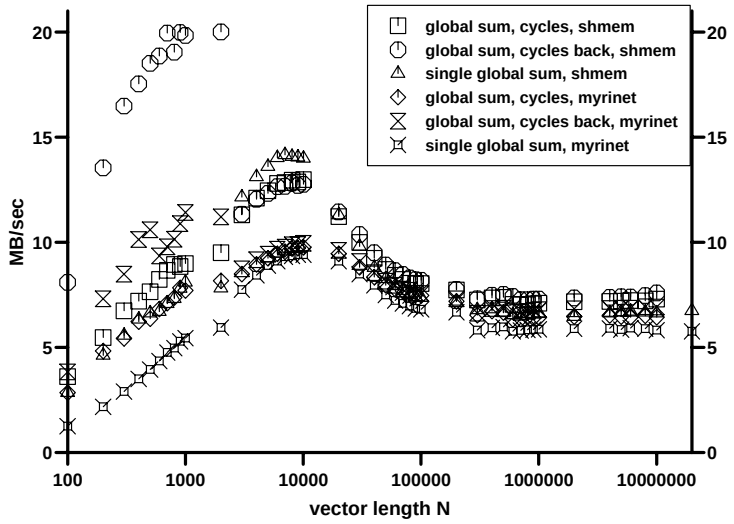

Figure 22: Combine rates for global sum to one processor, all methods, USED_PROCS=32, Double Precision vectors, myrinet means that myrinet is used for all communications, shared-memeory means that within each node shared-memory is used.

Figures 23 and 24 only show the combine rates measured with cycles forward and with single combine as the rates with cycles backward deliver wrong results for $N \leq 2000$ with numbers of processors greater than 5 and almost the same results as cycles forward for $N \geq 3000$.

The steps in timing when another stage is added for global sum, i.e. between 4 and

5 processors, between 8 and 10 processors, and between 16 and 20 processors are very small, although an additional computation step is there, too.

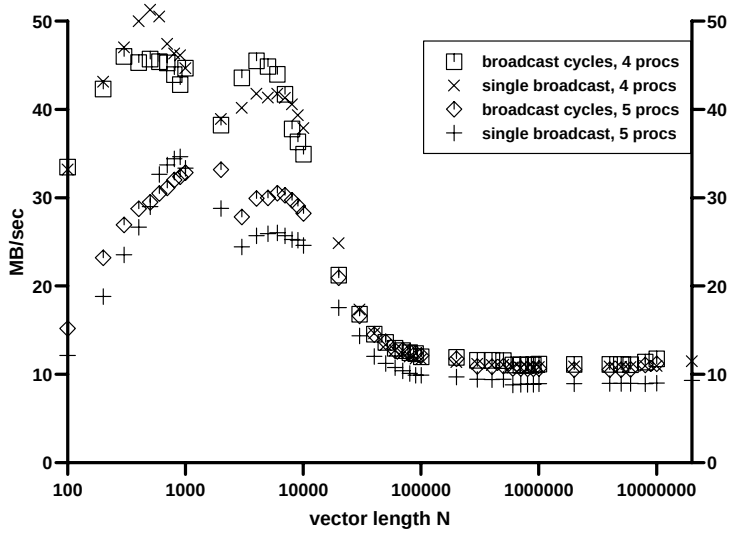

Figure 23: Combine rates for global sum to one processor, 4 and 5 processors taking part, shared-memory communication is used within one node.

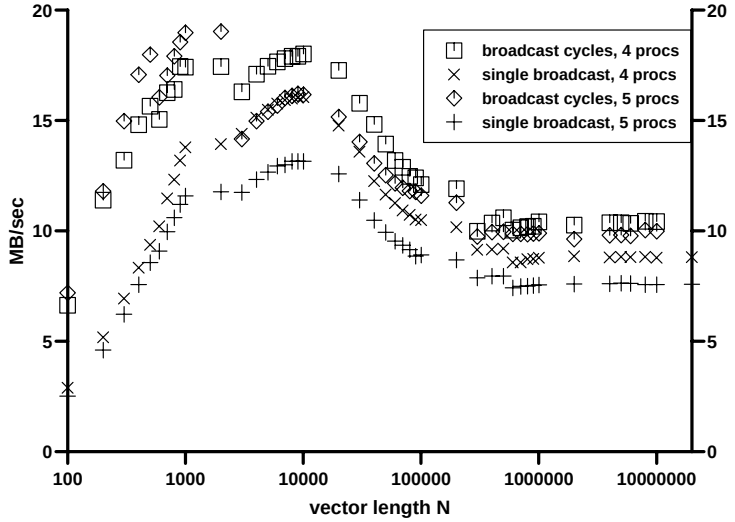

Figure 24: Combine rates for global sum to one processor, 4 and 5 processors taking part, myrinet communication is used between all processors.

5 Combine Sum to Broadcast Ratio

For tuning ScaLAPACK 1.7 and especially for the level 3 PBLAS (release 2) contained in it the ratio of the transfer cost per element for the combine sum to one processor and the broadcast operation (called CBRATIO and set by a DEFINE statement) plays an important role.
We computed this ratio for all message sizes, processor numbers, all kinds of measurement, and both kinds of communication. The measurement of single broadcasts and single global combines are not reliable and the ratios computed with these values differ significantly from the other two kinds of measurement. Thus we omitted these values in the following figures.

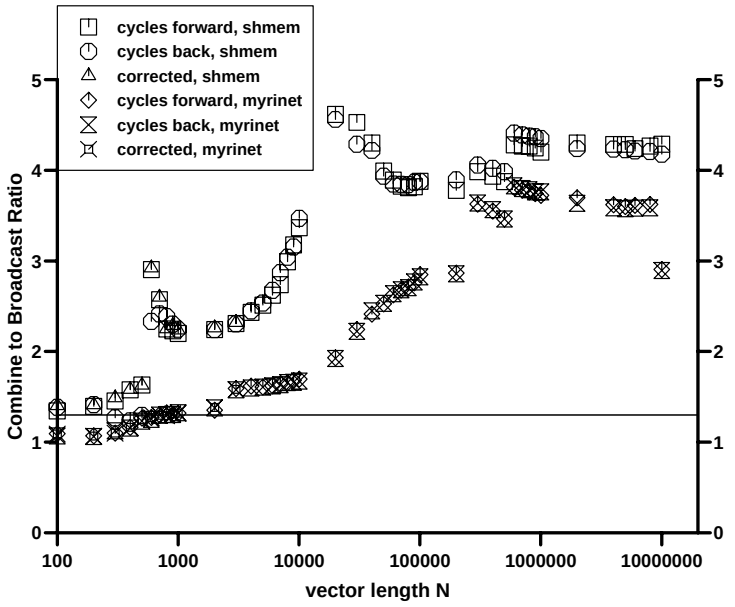

Figure 25: Combine to broadcast ratio, only two processors are taking part in the communication. ”Corrected” means the ratio: time for global combine sum to one processor measured with combine cycles forward to time for broadcast measured with cycles backward. The ”corrected” ratio is only shown for $N \leq 3000$. The horizontal line shows the default ratio choosen in ScaLAPACK.

For small messages up to 3000 Double Precision values none of the ways to measure execution times delivers reliable results. This leads to the problem that for those messages no ratio can be identified. As the most reliable results for small messages were achieved with broadcasts in a cycle backward and global sum in a cycle forward we also computed a ”corrected” result by dividing those times. When the messages become larger there is almost no difference in the times measured for broadcast in a cycle forward and backward, thus the corrected ratio is the same as the one for both measurements done in cycles forward. So we will only

show the corrected ratio for $N \leq 3000$.
For two processors both kinds of cycles are the same so the results should not differ very much, and this is true as can be seen from figure 25.
With only two processors taking part in the communication the ratio is larger than the default value for all messages longer than 2000 Double Precision words, in this case the value should be chosen higher. But this is the most unusual case as for a parallel program at least four processors will normally be used. Using four processors the ratio is already less than 1.3 for messages smaller than 1000 Double Precisions using shared-memeory and for less than 20000 Double Precision values using myrinet (see figure 26).

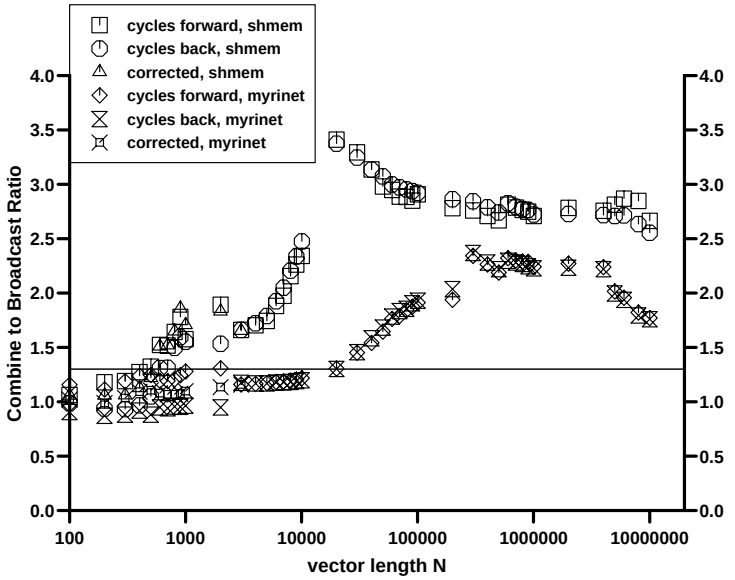

Figure 26: Combine to broadcast ratio, four processors (1 node) are taking part in the communication. "Corrected" means the same as in figure 25. The horizontal line shows the default ratio choosen in ScaLAPACK.

As expected there is an increase of the ratio for $N > 10000$ but not only using myrinet communication even within a node but also using shared-memeory on each node. Thus there is a problem to decide how long the messages will be on average and how many processors will be used and whether shared-memeory is used within each node or not.
Unfortunately the code of the parallel BLAS 3 routines is so complicated that we could not find out how the ratio really influences the execution. Thus it is not clear whether the broadcast to combine ratio for blocks of $MB \times KB$ where MB is the row block size and KB the column block size of a matrix is important or whether complete local parts of matrices consisting of several blocks are to be broadcast or combined.
In the former case the ratio for small matrices up to $100 \times 100 = 10000$ is

important, in the latter case the ratio for large matrices should be taken, which is impossible as the ratio varies too much for $N \geq 20000$. The steepest increase of the ratio for both kinds of communication and $NP \geq 5$ is between $N = 20000$ and $N = 300000$, which seems to be the most interesting region.
In all cases using more than one node the behaviour is similar. Figure 27 shows the combine to broadcast ratio using 12 processors as an example.

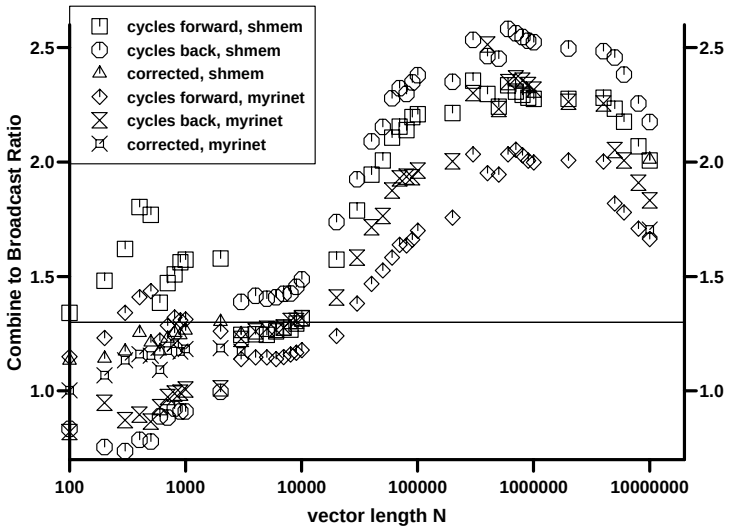

Figure 27: Combine to broadcast ratio, 12 processors (3 nodes) are taking part in the communication. Corrected means the same as in figure 25. The horizontal line shows the default ratio choosen in ScaLAPACK.

The ratio increases from 1.19 for $N = 10000$ to 1.70 for $N = 200000$ ($NP = 10$) or 1.26 to 1.97 ($NP = 5$) using myrinet communication even within a node and measuring in forward cycles. Measuring in cycles backward they increase from 1.31 for $N = 10000$ to 1.98 for $N = 200000$ with 10 processors and from 1.40 to 2.18 with 5 processors. Using shared-memeory within each node gives still other values: 1.30 to 2.00 using 20 processors and 1.71 to 2.78 using 5 processors and measuring cycles forward, 1.47 to 2.28 using 14 processors and 1.99 to 2.86 using 5 processors and measuring in cycles backward. The values are the smallest and the largest values for a given message length and kind of communication and measurement.
From $N = 200000$ to $N = 2000000$ the ratios remain almost constant and then they decrease again almost as fast as they increase between $N = 10000$ and $N = 200000$. The values for $N \geq 1000000$ are not interesting for the combine to broadcast ratio because this would mean that matrices of more than 1000×1000 would be broadcast or combined. Usually the local pieces will not be larger than this as for some parallel BLAS routines additional buffers of the same size as the messages to be sent are allocated and this means too much additional storage.

In all cases with more than 8 processors and the number of processors not being a power of two there is a large difference between the ratio measured with cycles forward and the ratio measured with cycles backward. This difference is almost as large as the difference between the communication methods shared-memeory or myrinet within one node.

Taking into account all the difficulties in measuring the combine to broadcast ratio it can be said that any value between 1.2 and 2.0 is plausible and only measurements of the performance of PBLAS routines could give further indication of how to choose this ratio. We even believe that if we compiled the PBLAS library with different values for CBRATIO there will be sizes and configurations where one value leads to better performance and cases where another value leads to better performance. Thus we decided not to change the default.

References

[1] *ZAMpano - ZAM Parallel Nodes*
http://zampano.zam.kfa-juelich.de

[2] *BLACS - Basic Linear Algebra Communication Subprograms*
http://www.netlib.org/blacs/index.html

[3] J.J. Dongarra, R.C. Whaley, *LAPACK working Note 94, A User's Guide to the BLACS v1.1*
http://www.netlib.org/lapack/lawns/index.html

[4] *ScaLAPACK*
http://www.netlib.org/scalapack/index.html

[5] *ATLAS - Automatically Tuned Linear Algebra Software*
http://www.math-atlas.sourceforge.net/

[6] T. Oechtering, I. Gutheil, *Performance Measurements of BLACS Routines on CRAY T3E*, Interner Bericht FZJ-ZAM-IB-2000-06, Juli 2000